

Corso di Ingegneria del Software – Appello straordinario 2 Novembre 2016

V. Gervasi, L. Semini Dipartimento
di Informatica, Università di Pisa, a.a. 2015/16

Nome: _____

Cognome: _____

Matricola: _____

Corso: A B

Con riferimento al caso di studio *REBU* presentato durante il corso, si considerino:

- la prenotazione di una corsa;
- la possibilità di prenotare da/per un aeroporto, associando il numero del volo alla prenotazione.

Il servizio di prenotazione di *REBU* si arricchisce di una nuova feature: la possibilità di associare un numero di volo a una prenotazione, in caso di corse da/per un aeroporto.

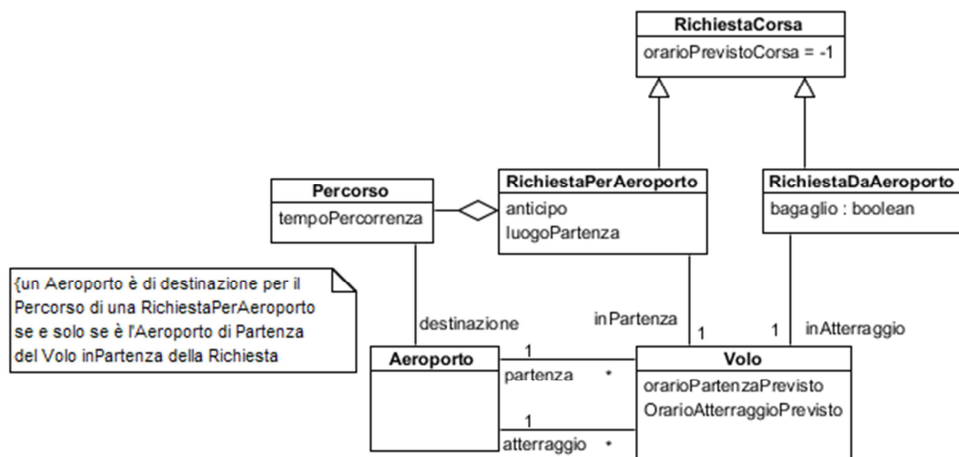
Nel caso di spostamenti verso un aeroporto, al momento della richiesta, il cliente specifica il numero del volo e con quanto anticipo, in minuti, desidera arrivare in aeroporto rispetto alla partenza del volo. Il sistema si interfaccia con i servizi dell'aeroporto e ottiene l'orario di partenza. Si interfaccia quindi con un servizio di calcolo di percorso stradale (come quello offerto da Google Maps) per ottenere i tempi di percorrenza tra la posizione di ritrovo e l'aeroporto, e stabilire un orario di partenza per la corsa. A questo punto procede come con una normale prenotazione.

Nel caso di spostamenti da un aeroporto, al momento della richiesta il cliente specifica il numero del volo e se avrà del bagaglio imbarcato da attendere dopo l'atterraggio. Il sistema si interfaccia con i servizi dell'aeroporto, ottiene l'orario previsto di atterraggio del volo, stabilisce un orario di partenza per la corsa, e procede come con una normale prenotazione.

Nelle ultime due ore prima dell'inizio della corsa associata a un volo, *REBU* monitora eventuali ritardi dei voli. In caso di ritardo superiore ai 15 minuti, ricalcola l'orario di partenza della corsa (ricalcolando anche i tempi di percorrenza che potrebbero essere cambiati), cerca un nuovo autista per la corsa ritardata (stesso tipo di auto, se superiore senza aggravio per l'utente), e cancella la precedente prenotazione. Quindi informa l'utente del nuovo orario.

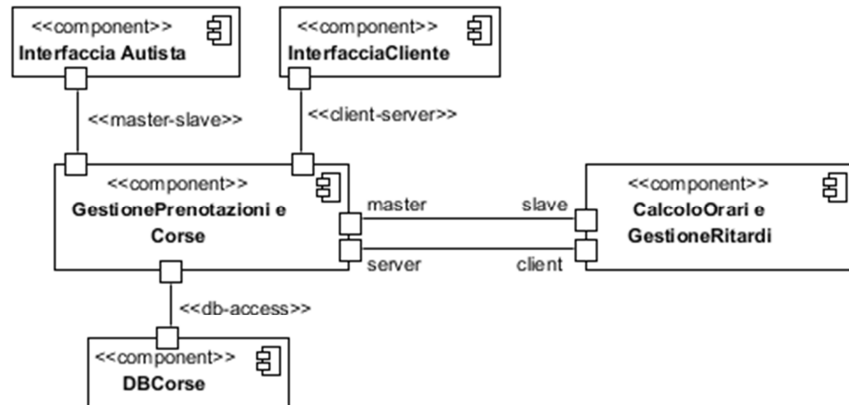
Domanda 1 . Dare la narrativa del caso d'uso “prenotazione di una corsa verso l'aeroporto”.

Domanda 2. Dato il seguente diagramma delle classi, che descrive le entità di interesse per i nuovi servizi di prenotazione, dare due possibili (e semplici!) diagrammi degli oggetti.



Domanda 3. Dare il diagramma di macchina a stati che descrive l'evoluzione degli oggetti di tipo RichiestaPerAeroporto.

Domanda 4. La progettazione architetturale ha portato alla seguente architettura (vista C&C), e in particolare all'individuazione di una componente CalcoloOrari e GestioneRitardi che si occupa di calcolare l'ora di inizio di una corsa e di monitorare eventuali ritardi dei voli e, se necessario, modificare l'orario di partenza. Si comporta da slave nei confronti della componente GestionePrenotazioni e Corse nel primo caso, da cliente nel secondo.



Inoltre, i servizi offerti dai vari aeroporti non sono uniformi: alcuni offrono la possibilità di abbonarsi alle informazioni su un volo e informano REBU di eventuali ritardi (modalità PUSH), altri offrono solo un servizio di informazioni domanda/risposta (modalità PULL).

Dare un diagramma di struttura composita della componente CalcoloOrari e GestioneRitardi.

Domanda 5. Il metodo `calcolaPartenzaPerAeroporto` calcola l'orario di inizio di una corsa, conoscendo, l'anticipo, il tempo di percorrenza (che si assume qui indipendente dall'orario del volo), e il volo. Inoltre, restituisce un nuovo orario di partenza in caso di scostamenti superiori ai 15 minuti rispetto a un orario precedentemente calcolato.

Si consideri il seguente frammento di codice, dove il valore del parametro `orarioPrevistoCorsa` è -1 quando non è ancora stato calcolato un orario di partenza, diverso altrimenti:

```

private int calcolaPartenzaPerAeroporto(int orarioPrevistoCorsa, int
anticipo, int tempoPercorrenza, Volo v){
    int o = v.getOrarioPartenza();
    int d = o - anticipo - tempoPercorrenza ;
    int t = d - now();

    if (t > 120 || orarioPrevistoCorsa == -1)
        return d;
    if (t<0)
        return -5;
    int r = d - orarioPrevistoCorsa;
    if (Math.abs (r)>15)
        return d;
    else
        return orarioPrevistoCorsa;
}
  
```

Dare il diagramma di flusso del metodo e un insieme minimo di casi di test per avere copertura delle decisioni e copertura delle condizioni.